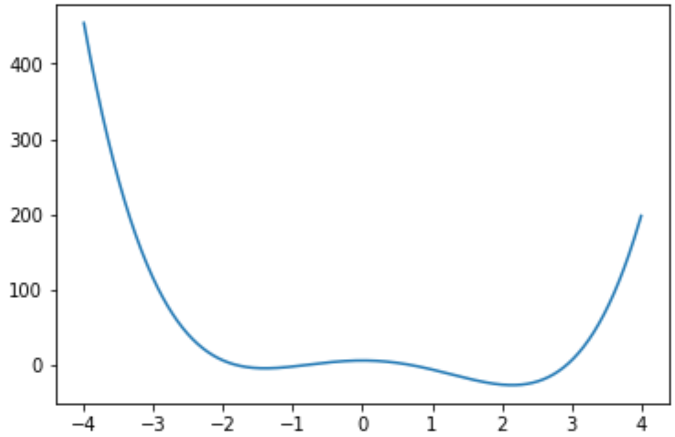
Please refer apt321_ss11381_ML_HW4_PART1_Source_Code.ipynb file for Source Code

Question 1:
Graph function for x in the interval [-4,4]



a.
Solution:
```
Local Minima at x = -1.3971679687499976
Global Minima at x = 2.1471808637959735
```

b.
Solution:
Setting x = −4 and η = 0.001

Running 6 Iterations:
```
Before entering the iteration, x is: -4, f(x) is: 454

Iteration 1: X = -3.488, f(x) = 240.90741220147203
Iteration 2: X = -3.159231053824, f(x) = 148.52441854620668
Iteration 3: X = -2.9229164225026394, f(x) = 99.4029877988204
Iteration 4: X = -2.742031675863951, f(x) = 70.0712149441725
Iteration 5: X = -2.59779507407776, f(x) = 51.16573699678776
Iteration 6: X = -2.4794003442716166, f(x) = 38.29644231132754

The minimum occurs at -2.4794003442716166
```

Running 1200 Iterations:
```
Before entering the iteration, x is: -4, f(x) is: 454

X converges at Iteration 250
Iteration 1195: X = -1.3971808598447308, f(x) = -4.348957724100302
Iteration 1196: X = -1.3971808598447308, f(x) = -4.348957724100302
Iteration 1197: X = -1.3971808598447308, f(x) = -4.348957724100302
Iteration 1198: X = -1.3971808598447308, f(x) = -4.348957724100302
Iteration 1199: X = -1.3971808598447308, f(x) = -4.348957724100302
Iteration 1200: X = -1.3971808598447308, f(x) = -4.348957724100302

The minimum occurs at -1.3971808598447308
```

The value of x has converged to local minimum.

c.
Solution:
Setting start with x = 4

Running 6 Iterations:

```
Before entering the iteration, x is: 4, f(x) is: 198

Iteration 1: X = 3.68, f(x) = 110.61233152000005
Iteration 2: X = 3.450886144, f(x) = 64.53629857986431
Iteration 3: X = 3.276396901609702, f(x) = 37.31076190742675
Iteration 4: X = 3.138067975365072, f(x) = 19.971643359608052
Iteration 5: X = 3.0252501730040535, f(x) = 8.322601113072949
Iteration 6: X = 2.9312689375235244, f(x) = 0.17557478693807127

The minimum occurs at 2.9312689375235244
```

Running 1200 Iterations:

```
Before entering the iteration, x is: 4, f(x) is: 198

X converges at Iteration 170
Iteration 1195: X = 2.1471808598447315, f(x) = -26.611979775899705
Iteration 1196: X = 2.1471808598447315, f(x) = -26.611979775899705
Iteration 1197: X = 2.1471808598447315, f(x) = -26.611979775899705
Iteration 1198: X = 2.1471808598447315, f(x) = -26.611979775899705
Iteration 1199: X = 2.1471808598447315, f(x) = -26.611979775899705
Iteration 1200: X = 2.1471808598447315, f(x) = -26.611979775899705

The minimum occurs at 2.1471808598447315
```

The value of x has converged to global minimum.

d.
Solution:
Setting x = −4 and η = 0.01

Running 1200 Iterations:

```
Before entering the iteration, x is: -4, f(x) is: 454

Iteration 1: X = 1.12, f(x) = -8.71561728
Iteration 2: X = 1.35166976, f(x) = -14.187225687602176
Iteration 3: X = 1.588129914065571, f(x) = -19.554356180837104
Iteration 4: X = 1.8001695002820235, f(x) = -23.55150883046352
Iteration 5: X = 1.9599549783032466, f(x) = -25.64204722189585
Iteration 6: X = 2.0585082124451546, f(x) = -26.383081197323108
X converges at Iteration 18
Iteration 1195: X = 2.147180859844728, f(x) = -26.611979775899698
Iteration 1196: X = 2.147180859844728, f(x) = -26.611979775899698
Iteration 1197: X = 2.147180859844728, f(x) = -26.611979775899698
Iteration 1198: X = 2.147180859844728, f(x) = -26.611979775899698
Iteration 1199: X = 2.147180859844728, f(x) = -26.611979775899698
Iteration 1200: X = 2.147180859844728, f(x) = -26.611979775899698

The minimum occurs at 2.147180859844728
```

The value of x has converged to global minimum in early iteration as compared to (c), because the le arning rate is high (η = 0.01).

e.
Solution:
Setting x = −4 and η = 0.1

Running 100 Iterations:

```
Before entering the iteration, x is: -4, f(x) is: 454

Iteration 1: X = 47.2, f(x) = 9689505.955200002
Iteration 2: X = -82626.05440000002, f(x) = 9.321875746621314e+19
Iteration 3: X = 451278842347294.06, f(x) = 8.294875771953852e+58
Iteration 4: X = -7.352328532672759e+43, f(x) = 5.8442611657954e+175
Iteration 5: X = -inf, f(x) = nan
```

X value is bouncing all over from positive to negative and never converges on a single point. This is because learning rate is too high (η = 0.1).

Question 2:

Solution (2)

(a) The pseudocode in fig 11.11 is performing stochastic gradient descent where the weight v_{ih} for $i=2, h=1$ is updated once for every training samples analyzed.

There are 500 samples & the algorithm runs on all the 500 samples for every of 100 epochs, so total no. of updates –

$$500 \times 100 = 50000 \text{ updates.}$$

(b) (i) $i=2, h=3$

$$E_{new}(w, v|x) = \frac{1}{2}\left[3(r_1-y_1)^2 + 7(r_2-y_2)^2\right]$$

$$\therefore \Delta v_{ih} = \Delta v_{23} = -\eta \frac{\partial E(w,v|x)}{\partial v_{23}}$$

$$= -\eta \frac{\partial}{\partial v_{23}}\left[\frac{1}{2}\left[3(r_1-y_1)^2 + 7(r_2-y_2)^2\right]\right]$$

$$= -\eta \frac{\partial}{\partial v_{23}}\left[\frac{1}{2}\left[7(r_2-y_2)^2\right]\right]$$

$$= -7\eta(r_2-y_2)\left[\frac{-\partial}{\partial v_{23}} y_2\right]$$

$$\therefore \Delta v_{23} = 7\eta(r_2-y_2)\frac{\partial}{\partial v_{23}}(v_2^T z) = 7\eta(r_2-y_2)z_3 \quad //$$

(ii) $$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$= -\eta \sum_t \frac{\partial E^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_h^{(t)}} \cdot \frac{\partial z_h^{(t)}}{\partial w_{hj}}$$

$$\therefore \Delta w_{hj} = -\eta \sum_t \frac{\partial E^t}{\partial y^{(t)}} v_h z_h^{(t)} (1 - z_h^{(t)}) x_j^{(t)}$$

$$\therefore \frac{\partial E^{(1)}}{\partial y^{(1)}} = -3(r^{(1)} - y^{(1)})$$

$$\frac{\partial E^{(2)}}{\partial y^{(2)}} = -7(r^{(2)} - y^{(2)})$$

$$\therefore \Delta w_{hj} = -\eta \Big[\left[-3(r^{(1)} - y^{(1)}) v_h z_h^{(1)} (1 - z_h^{(1)}) \cdot x_j^{(1)} \right] + \left[-7(r^{(2)} - y^{(2)}) v_h z_h^{(2)} (1 - z_h^{(2)}) \cdot x_j^{(2)} \right] \Big]$$

$$= \eta \left[3(r^{(1)} - y^{(1)}) v_h z_h^{(1)} (1 - z_h^{(1)}) \cdot x_j^{(1)} + 7(r^{(2)} - y^{(2)}) v_h z_h^{(2)} (1 - z_h^{(2)}) \cdot x_j^{(2)} \right]$$

Question 3:

a.
Solution:
As NeuralNetRK uses a linear function, it can output values that are negative numbers.
NeuralNetCB, NeuralNetCK and NeuralNetRZeroOne cannot produce negative output.

b.
Solution:
Here, only NeuralNetCK ensures that the sum of the outputs $y_1, \ldots, y_k$ will be 1.

c.
Solution:
Here, K=3 (p_1, p_2 and p_3 -> Probabilities of class face, cat and tree).
Therefore, it would be appropriate to use NeuralNetCK, since K > 2 classes and the sum of the outputs p_1, p_2, p_3 is 1.

d.
Solution:
NeuralNetZeroOne is the appropriate option as:

1. It is a classification problem and there are two values 0/1.
2. NeuralNetCB cannot applied as it is a single output solution.
3. Problem expects probabilities for politics and style.
   Eg. if politics: x_1=1, else 0, if formal_style: x_2=1, else 0

Question 4:

a.
Solution:
Neural Net Algorithm learns from weight which is not appropriate in this case.
Example consider - almond = 1, anise = 2, creosote = 3, and fishy = 4:
Here, neural net algorithm will consider the 'fishy' value as weighted value(Prediction = weight * odor) 4 (double of anise) , which is not true.

Whereas in case of Random Forest, it will support rules such as:

```
if odor = 1:
    //encoding of almond
    process algo…
else if odor = 2:
        //encoding of anise
        Process algo…
```

Therefore, if an algorithm is learning by weight, we should not use label encoding and hence, in our case, it would be fine to do this if we were using a random forest, rather than a neural net. An alternative solution would be to use one-hot encoding.

b.
i. Solution:
For transforming attribute ‘stalk shape`, we can add one-hot encoding in our existing transformed dataset: if stalk_shape=tapering: z_5=1, else 0; if stalk_shape=enlarging: z_6=1, else 0,

Transformed Dataset:

	z_1	z_2	z_3	z_4	z_5	z_6	label
x_1	0	0	0	1	1	0	0
x_2	0	0	1	0	0	1	0

ii. Solution:
If we use one-hot encoding, there is a problem of losing the hierarchy (high > medium > low) and hence, it is better to use label encoding which retains the order.

iii. Solution:
Here, it is appropriate to use (0,1) instead of one-hot encoding as it will generate the same dataset.

Example: Dataset for Coin Tosses.

Attributes	Outcome
x_1	Heads
x_2	Tails
x_3	Heads

Use (0,1) encoding -> if heads: x=1, if tails: x=0

Attributes	Outcome
x_1	1
x_2	0
x_3	1

Using one-hot encoding will give

Attributes	z_1=Heads	z_2=Tails
x_1	1	0
x_2	0	1
x_3	1	0

Here, z_2 is redundant and can be eliminated as z_1 can alone represented the whole dataset.

In case of three attributes, the value of attribute can be either of 3 outcomes and hence, it is appropriate to use one-hot encoding to represent the data.